Listwise Recommendation Approach with Non-negative Matrix Factorization

Gaurav Pandey[1] and Shuaiqiang Wang[2]

[1] University of Jyvaskyla, Finland,
gaurav.g.pandey@jyu.fi,
[2] Data Science Lab, jd.com, China

Abstract. Matrix factorization (MF) is one of the most effective categories of recommendation algorithms, which makes predictions based on the user-item *rating matrix*. Nowadays many studies reveal that the ultimate goal of recommendations is to predict correct rankings of these unrated items. However, most of the pioneering efforts on *ranking-oriented* MF predict users' item ranking based on the original rating matrix, which fails to explicitly present users' preference ranking on items and thus might result in some accuracy loss. In this paper, we formulate a novel listwise user-ranking probability prediction problem for recommendations, that aims to utilize a *user-ranking probability matrix* to predict users' possible rankings on all items. For this, we present *LwRec*, a novel listwise ranking-oriented matrix factorization algorithm. It aims to predict the missing values in the user-ranking probability matrix, aiming that each row of the final predicted matrix should have a probability distribution similar to the original one. Extensive offline experiments on two benchmark datasets against several state-of-the-art baselines demonstrate the effectiveness of our proposal.

Keywords: Recommender systems, Collaborative Filtering, Ranking

1 Introduction

Conventional recommendation algorithms like collaborative filtering follow a rating-oriented paradigm. They generally learn a recommendation model with users' observed historical ratings, using which they predict users' ratings on their unrated items. Nowadays, ranking-oriented recommender systems are receiving increasing attention from both academic communities and industry. Many studies reveal that the ultimate goal of recommendations is to predict correct rankings of these unrated items, and prediction of accurate ranking is more important than predicting accurate rating scores [1, 2]. Accurate prediction of ratings does not necessarily imply improvement in the ranking results.

To elaborate, let us consider items: $\{a, b, c\}$ with their correct ratings R : $\{5, 4, 3\}$ and two rating predictions $P_1 : \{3, 4, 5\}$ and $P_2 : \{3, 2, 1\}$. A rating oriented approach would prefer P_1 over P_2, since predicted ratings in P_1 are more accurate, being closer to the ratings in R. However, the order in P_1 $(a < b < c)$

is completely opposite to the desired order ($a > b > c$). In contrast, a ranking oriented approach would prefer P_2, as it predicts the correct ranking, i.e. ($a > b > c$). This would generate desirable results (correct order of items), in spite of having lower accuracy in predicted ratings.

Given the above argument, some pioneering efforts on *ranking-oriented* recommendation algorithms have been proposed. Due to the effectiveness of matrix factorization (MF) algorithms in rating-oriented recommender systems, a few ranking-oriented MF algorithms have been presented, reporting state-of-the-art results. However, most of the pioneering efforts on *ranking-oriented* MF predict users' item ranking based on rating scores, failing to explicitly present users' preference ranking on items and thus possibly resulting in some accuracy loss.

Therefore, we define a novel listwise user-ranking probability prediction problem for recommendations. We utilize the *listwise user-ranking probability matrix* [3] to explicitly characterize users' preference on items. Given a set of rating scores on items, each ranking on items might be possible, where "correct" rankings (higher scores are ranked at top positions) receive greater probabilities. Thus for each user, the probabilities on users' all possible rankings could formulate as a user-ranking probability matrix, where each element presents a probability that certain user holds certain ranking on items. Thus each row of the user-ranking probability matrix consists of users' probabilities on different item rankings, forming a distribution. With the initial probabilities of users' possible rankings on *their rated items*, the listwise user-ranking probability prediction problem aims to predict probabilities of users' possible rankings on *all items*. Meanwhile, the predictions should satisfy the requirements of probability distributions: each element in the probability matrix should be between 0 and 1, and sum of each row should be 1.

Given a collection of items, there might be a very large number of possible rankings (i.e. $n!$ rankings for n items), resulting in a extremely huge ranking probability matrix and calculations in training. In this study, we only consider the top-k ranked items in rankings, and the size of the matrix could be shrinked significantly, especially the size of the ranking probability matrix is equal to that of the user-item rating matrix when $k = 1$. Based on this matrix, we then present *LwRec*, a novel listwise ranking-oriented MF algorithm, which minimizes the difference between the initial distribution on the known rankings and the final distribution on all items with predictions for each user. Considering the non-negative property for each element, we adapt non-negative MF to implement *LwRec*. Our experimental results on benchmark datasets demonstrate significant performance gains over state-of-art recommender algorithms.

To summarize, our contributions are as follows. (1) We define a novel listwise user-ranking probability prediction problem for recommendations. (2) We present an effective algorithm to solve the problem based on non-negative MF. (3) We achieve significant performance gains against state-of-the-art recommendation algorithms on benchmark datasets.

The rest of the paper is organized as follows. Section 2 briefly presents the related work and Section 3 describes the problem formulation. Then, we explain

the *LwRec* approach in Section 4 followed by experimental setup in Section 5. Finally, Section 6 presents the results and Section 7 concludes the paper.

2 Related Work

This section presents related work for collaborative filtering (CF) recommendation algorithms, which use only the ratings given by the users for the items, and do not need the domain knowledge. They are mainly of two types: rating oriented and ranking oriented. While rating oriented algorithms predict unknown item ratings for each user, ranking oriented algorithms predict item rankings. Both of them can be further categorized as memory-based or model based.

Rating Oriented Algorithms: Memory-based rating oriented algorithms are either user-based CF [4], that utilize similarities between users on the basis of available ratings; or item-based CF [5], that utilize similarities between items. Various advanced versions of this approach have been introduced. For example, SLIM [6] directly learns from the data, a sparse matrix of aggregation coefficients that are analogous to the traditional item-item similarities. FISM [7] learns the item-item similarity matrix as a product of two low-dimensional latent factor matrices. Model-based rating oriented algorithms aim to predict ratings by learning a model from observed ratings. Traditional model of this type is matrix factorization (MF) [8], that uses dimensionality reduction to decrease the distance between predicted and observed rating matrices. Some of the models that are based on matrix factorization are: Probabilistic MF [9], Non-negative MF [10], Factorization Machines [11], Hierarchical Poisson MF [12] and LLORMA [13].

Ranking Oriented Algorithms: EigenRank [14] is a well known ranking oriented memory based CF algorithm that follows the pairwise approach. It employs a greedy aggregation method to aggregate predicted pairwise preferences of items into total ranking. VSRank [15] represents users' pairwise preferences for items by using vector space model and utilizes the relative importances of each pairwise preference. Moreover, various model-based ranking oriented CF algorithms have been introduced that try to optimize a ranking oriented objective function. Some of the notable algorithms of this type are: CLiMF [16], CoFiRank [17], ListCF [3] and GBPR [18].

3 Problem Formulation

3.1 User-Ranking Probability Matrix

Considering m users and n items, for each user there are obviously $n!$ possible rankings of items. Given a set of rating scores on items, each ranking on items might be possible, where "correct" rankings (higher scores are ranked at top positions) receive greater probabilities. The probability of item rankings could be derived with the Plackett-Luce model [19], which is a widely used permutation

(each permutation is actually a ranking) probability model in various domains. Each ranking ρ can be represented as an ordered list $(\rho_1, \rho_2 \dots \rho_n)$, where ρ_i represents the item at the ith position, and positions of the items are unique. Hence, the probability of the ranking ρ can be calculated as:

$$Prob(\rho) = \prod_{i=1}^{n} \frac{\gamma(r_{\rho_i})}{\sum_{j=i}^{n} \gamma(r_{\rho_j})}, \tag{1}$$

where r_{ρ_i} is the rating for the item ρ_i and $\gamma(r) = e^r$.

Since, there are $n!$ rankings of items, which is a large number of rankings even for a small value of n, it makes the computation impractical. Hence, we employ the same approach as Huang et al. [3], that uses an alternative efficient method introduced by Cao et al. [20]. The approach focuses only on top k items in the rankings, leading to $\frac{n!}{(n-k)!}$ different top k sets. So, the probability of the rankings ρ_S whose top-k items are exact $S = \{i_1, i_2 \dots i_k\}$ can be calculated as:

$$Prob(\rho_S) = \prod_{j=1}^{k} \frac{\gamma(r_{i_j})}{\sum_{l=j}^{n} \gamma(r_{i_l})} \tag{2}$$

We have m users and for each user we have $p = \frac{n!}{(n-k)!}$ ranking sets for top k items. Now, we construct the user-ranking probability matrix $\Theta_{m \times p}$. In Θ, each row corresponds to a particular user and contains the probabilities for the p rankings. To clarify, if $Prob_{u_i}(S_j)$ represents the probability of ranking S_j calculated for the user u_i (where $1 \leq j \leq p$ and $1 \leq i \leq m$), then:

$$\begin{aligned} \Theta_{i,j} = & Prob_{u_i}(S_j), \text{if ratings of all items in } S_j \text{ are known,} \\ & \perp, \text{otherwise} \end{aligned} \tag{3}$$

Especially when $k = 1$, i.e. when we consider only the top-1 items in all rankings, the size of Θ is equal to that of the user-item rating matrix. This is because, in this case, $p = \frac{n!}{(n-1)!} = n$.

3.2 Objective and Constraints

Given the matrix of known top k probabilities of items: $\Theta_{m \times p}$, where $p = \frac{n!}{(n-k)!}$, we aim to predict the unknown probabilities, that in turn can be used to generate recommendations. This can be achieved by using a listwise loss function and optimizing it using matrix factorization. For this, we define the following objective and the two related constraints:

Objective: Using two matrices $U_{z \times m}$ and $G_{z \times p}$ that construct the predicted probability matrix $U^\top G$, utilize a listwise loss function and matrix factorization to minimize the distance between Θ and $U^\top G$.

C1: Values in $U^\top G$ should be in the range 0 to 1 (as they are probabilities). i.e. $0 \leq U_{ij} \leq 1, \forall i = 1 \dots z$ and $\forall j = 1 \dots m$.

C2: Sum of each row of $U^\top G$ should be 1 (as a row contains probabilities of rankings for a particular user, that should sum up to 1). i.e. $\sum_{j=1}^{p} (U^\top G)_{ij} = 1, \forall i = 1 \dots m$.

Definition 1 (Listwise User-Ranking Probability Prediction). *Given a user-ranking probability matrix Θ, where each observed element of $\Theta_{i,p}$ indicates certain user's probability for her certain (top-k) preference ranking on her rated items, and each row of Θ forms a probability distribution. The listwise user-ranking probability prediction problem aims to predict each user's probability of her top-k preference ranking on all items, where each row of $U^\top G$ forms a probability distribution as well after prediction, and each user's two distributions, observed and predicted, should be as similar as possible. Formally,*

$$\begin{aligned} &\arg\min_{U,G} \sum_{i=1}^{m} \mathit{diff}\,(\Theta_i, (U^\top G)_i), \\ &\quad \text{s. t. } 0 \le (U^\top G)_{ij} \le 1, i = 1, 2, \ldots, m, \text{ and } j = 1, 2, \ldots, p \text{ (C1)} \\ &\qquad \sum_{j=1}^{p} (U^\top G)_{ij} = 1, i = 1, 2, \ldots, m \text{ (C2)} \end{aligned} \tag{4}$$

Here $\mathit{diff}\,(\Theta_i, (U^\top G)_i)$ is the difference between two distributions Θ_i and $(U^\top G)_i$, i.e. the ith row of the user-ranking probability matrix before and after prediction.

4 Prediction Method

In this section, we present *LwRec* to solve our listwise user-ranking probability prediction problem. We use Kullback-Leibler divergence [21], a commonly used measure for calculating difference between probability distributions, to compute $\mathit{diff}\,(\Theta_i, (U^\top G)_i)$. In *LwRec*, we utilize non-negative matrix factorization (MF) [10] to implement our proposed algorithm, which can generate non-negative elements for U and G. Thus the elements of the user-ranking probability matrix $U^\top G$ are all non-negative. In order to satisfy the constraint C2, we introduce a collection of Lagrange penalty terms in the objective function $\sum_{j=1}^{p} (U^\top G)_{ij} = 1$ where $i = 1 \ldots m$. In standard Lagrange methods, the coefficients of Lagrange penalty terms could be either positive or negative, but in non-negative MF, all of the parameters have to be non-negative. Thus we introduce two non-negative vectors α and β, and regard $(\alpha_i - \beta_i)$ that can be either positive or negative, as the coefficient of the ith Lagrange penalty term to formulate our loss function. Moreover, addressing constraint C2 together with ensuring that the values in $U^\top G$ are non-negative, also satisfies constraint C1 (i.e. values in $U^\top G$ should be in range 0 to 1). The formulation of our loss function can be presented formally as follows:

$$\begin{aligned} L(U, G, \alpha, \beta) = &- \sum_{i=1}^{m} \sum_{j=1, \Theta_{ij} \neq \perp}^{p} \Theta_{ij} \log \frac{(U^\top G)_{ij}}{\sum_{l=1, \Theta_{il} \neq \perp}^{p} (U^\top G)_{il}} \\ &+ \sum_{i=1}^{m} (\alpha_i - \beta_i) \left(\sum_{j=1}^{p} (U^\top G)_{ij} - 1 \right) + \frac{\lambda_1}{2} ||U||^2 + \frac{\lambda_2}{2} ||G||^2, \end{aligned} \tag{5}$$

In Equation 5, the first term represents the main optimization objective from Equation 4, which defines the divergence between $U^\top G$ and observed probability matrix Θ. The second term is the weighted cumulative Lagrange penalty term

for constraint C2. The last two terms are l2-norms of U and G to avoid overfitting, where λ_1 and λ_2 are the respective coefficients. By expanding the *log* in $L(U, G, \alpha, \beta)$ and considering that the sum of each row in Θ is 1, the function can be reformulated as:

$$\begin{aligned} L(U,G,\alpha,\beta) = &\sum_{i=1}^{m} \log\Bigg(\sum_{l=1,\Theta_{il}\neq\perp}^{p} (U^\top G)_{il}\Bigg) - \sum_{i=1}^{m}\sum_{j=1,\Theta_{ij}\neq\perp}^{p} \Theta_{ij}\log\left((U^\top G)_{ij}\right) \\ &+ \sum_{i=1}^{m}(\alpha_i - \beta_i)\Bigg(\sum_{j=1}^{p}(U^\top G)_{ij} - 1\Bigg) + \frac{\lambda_1}{2}||U||^2 + \frac{\lambda_2}{2}||G||^2 \end{aligned} \tag{6}$$

To minimize the loss function using gradient descent, we compute its gradients with respect to the variables U, G, α and β and derive the following updates:

$$\begin{aligned} U_{ia} &\leftarrow U_{ia} - \eta_u\Bigg(\frac{\sum_{l=1,\Theta_{il}\neq\perp}^{p} G_{al}}{\sum_{l=1,\Theta_{il}\neq\perp}^{p}(U^\top G)_{il}} - \sum_{j=1,\Theta_{ij}\neq\perp}^{p}\frac{\Theta_{ij}G_{aj}}{(U^\top G)_{ij}} \\ &\quad + (\alpha_i - \beta_i)\sum_{j=1}^{p} G_{aj} + \lambda_1 U_{ia}\Bigg), \\ G_{aj} &\leftarrow G_{aj} - \eta_g\Bigg(\sum_{i=1}^{m}\frac{U_{ia}}{\sum_{l=1,\Theta_{il}\neq\perp}^{p}(U^\top G)_{il}} - \sum_{i=1,\Theta_{ij}\neq\perp}^{m}\frac{\Theta_{ij}U_{ia}}{(U^\top G)_{ij}} \\ &\quad + \sum_{i=1}^{m}(\alpha_i - \beta_i)U_{ia} + \lambda_2 G_{aj}\Bigg), \\ \alpha_i &\leftarrow \alpha_i - \eta_\alpha\Bigg(\sum_{j=1}^{p}(U^\top G)_{ij} - 1\Bigg) \text{ and } \beta_i \leftarrow \beta_i - \eta_\beta\Bigg(1 - \sum_{j=1}^{p}(U^\top G)_{ij}\Bigg) \end{aligned} \tag{7}$$

where, η_u, η_g, η_α and η_β are the step sizes. Now, using non-negative matrix factorization [10], we choose the step sizes such that:

$$\begin{aligned} \eta_u &= \frac{U_{ia}}{\frac{\sum_{l=1,\Theta_{il}\neq\perp}^{p} G_{al}}{\sum_{l=1,\Theta_{il}\neq\perp}^{p}(U^\top G)_{il}} + \alpha_i\sum_{j=1}^{p} G_{aj} + \lambda_1 U_{ia}}, \\ \eta_g &= \frac{G_{aj}}{\sum_{i=1}^{m}\frac{U_{ia}}{\sum_{l=1,\Theta_{il}\neq\perp}^{p}(U^\top G)_{il}} + \sum_{i=1}^{m}\alpha_i U_{ia} + \lambda_2 G_{aj}}, \\ \eta_\alpha &= \frac{\alpha_i}{\sum_{j=1}^{p}(U^\top G)_{ij}} \text{ and } \eta_\beta = \beta_i \end{aligned} \tag{8}$$

Substituting these values of the steps in updation formulas in Equations 7, we derive the following multiplicative updates:

$$\begin{aligned} U_{ia} &\leftarrow U_{ia}\frac{\sum_{j=1,\Theta_{ij}\neq\perp}^{p}\frac{\Theta_{ij}G_{aj}}{(U^\top G)_{ij}} + \beta_i\sum_{j=1}^{p} G_{aj}}{\frac{\sum_{l=1,\Theta_{il}\neq\perp}^{p} G_{al}}{\sum_{l=1,\Theta_{il}\neq\perp}^{p}(U^\top G)_{il}} + \alpha_i\sum_{j=1}^{p} G_{aj} + \lambda_1 U_{ia}}, \\ G_{aj} &\leftarrow G_{aj}\frac{\sum_{i=1,\Theta_{ij}\neq\perp}^{m}\frac{\Theta_{ij}U_{ia}}{(U^\top G)_{ij}} + \sum_{i=1}^{m}\beta_i U_{ia}}{\sum_{i=1}^{m}\frac{U_{ia}}{\sum_{l=1,\Theta_{il}\neq\perp}^{p}(U^\top G)_{il}} + \sum_{i=1}^{m}\alpha_i U_{ia} + +\lambda_2 G_{aj}}, \\ \alpha_i &\leftarrow \frac{\alpha_i}{\sum_{j=1}^{p}(U^\top G)_{ij}} \text{ and } \beta_i \leftarrow \beta_i\sum_{j=1}^{p}(U^\top G)_{ij} \end{aligned} \tag{9}$$

Algorithm 1: *LwRec* Algorithm

Input: Ratings for n items by m users, values k and z

Initialize $\Theta_{m \times p}$, where $p = \frac{n!}{(n-k)!}$ (See Equation 3)

Randomly initialize $U_{z \times m}, G_{z \times p}$, α_m and β_m with non-negative values

repeat

  Update U, G, α and β according to Equation 9

until *Reach convergence or the max iteration*;

return $U^\top G$

On optimizing U and G, The rows of $U^\top G$ would contain predicted probability distributions of top k rankings for users, that can be utilized to generate recommendations. Algorithm 1 summarizes our method.

5 Experimental Setup

5.1 Datasets

For our experiments, we use two MovieLens[3] data sets: MovieLens-100K and MovieLens-1M. MovieLens-100K dataset contains 100,000 ratings given by 943 users on 1682 movies. MovieLens-1M dataset is larger with 1,000,000 ratings given by 6040 users on 3952 movies. In MovieLens-100K as well as MovieLens-1M the ratings are given on an integer scale from 1 to 5. For both the datasets we assign 10 ratings for each user for testing and the rest for training.

5.2 *LwRec* Setup

For both datasets, we consider top 1 item rankings (i.e. $k = 1$), since the topmost position in a ranking is the most important one. Moreover, it also makes our experiments computationally inexpensive since when $k = 1$, $p = \frac{n!}{(n-1)!} = n$ (number of items). A higher value of k, would make the value of p huge. For example, for MovieLens-1M ($n = 3952$), when $k = 2$, $p \approx 1.56 \times 10^7$ and for $k = 3$, $p \approx 6.17 \times 10^{10}$. Probably higher values of k could result in some performance gains, but in this study we restrict the scope to experiment with $k = 1$. Moreover, for the matrices U and G, we have used the column length of 10 (i.e. $z = 10$).

We generate the probability distributions of known item rankings i.e. Θ using the training set and then generate the matrix of predicted probabilities i.e. $U^\top G$ (Algorithm 1). Since we use $k = 1$, each row of $U^\top G$ would contain probabilities for n items (as each item ranking has only one item in this case). Therefore, items in the test set can be simply ordered by their decreasing predicted probabilities.

5.3 Baselines

We used the following state-of-the-art algorithms as our comparison partners:

[3] http://grouplens.org/datasets/movielens/

1. CF: CF [22] calculates the similarity between users, and ranks the items according to the predicted ratings for each user.
2. Matrix Factorization (MF): User matrix U and item matrix I are optimized in MF [8], to minimize the difference between their product UI^T and rating matrix R. UI^T regenerates the rating matrix to predict unknown ratings.
3. EigenRank: EigenRank [14] is a pair-wise ranking-oriented algorithm that employs a greedy aggregation method to aggregate the predicted pairwise preferences of items into total ranking.
4. ListRankMF: ListRankMF [23] minimizes a loss function representing uncertainty between training and output lists produced by a MF ranking model.
5. FISM: Factored Item Similarity Models (FISM) [7] learn the item-item similarity matrix as a product of two low-dimensional latent factor matrices. While **FISMrmse** computes loss using sqaured error loss function, **FISMauc** considers a ranking error based loss function.
6. LLORMA: Local Low-Rank Matrix Approximation (LLORMA) [13] approximates the observed matrix as a weighted sum.
7. ListCF: ListCF [3], a ranking oriented CF algorithm, predicts item order for a user, based on similar users probability distributions over item permutations.

5.4 Evaluation Metrics.

We use the standard ranking accuracy metric called normalized discounted cumulative gain (NDCG@1-10) [24] that is able to handle multiple levels of relevance, to evaluate item rankings generated by *LwRec* and the baselines.

Statistical significance of observed differences between the performance of two runs is tested using a two-tailed paired t-test and is denoted using $^{\blacktriangle}$ (or $^{\blacktriangledown}$) for strong significance for $\alpha = 0.01$; or $^{\triangle}$ (or $^{\triangledown}$) for weak significance for $\alpha = 0.05$.

6 Results

In Table 1, we can see that *LwRec* outperforms the comparison partners for all the metrics (NDCG@1 to 10) for MovieLens-100K as well as MovieLens-1M. ListCF is the second best followed by LLORMA and FISMrmse, for both datasets. For MovieLens-100K, EigenRank and ListRankMF have comparable performances followed by MF and FISMauc. For MovieLens-1M, ListRankMF performs better than FISMauc followed by MF.

We also calculate statistical significance of *LwRec* against ListCF which is our best performing comparison algorithm. The results for MovieLens-100K show weak to strong statistical significance for most metrics and for MovieLens-1M the results have strong statistical significance in almost all cases.

7 Conclusion

In this paper, we defined a novel listwise user-ranking probability prediction problem. Then we described *LwRec*, a listwise recommendation algorithm, that

Table 1. Ranking Performance of *LwRec* against baselines

Statistical significance shown for *LwRec* against LLORMA

Performance for MovieLens-100K										
NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10
CF	0.5990	0.6394	0.6707	0.6938	0.7182	0.7442	0.7705	0.7970	0.8245	0.8546
MF	0.6629	0.6711	0.6918	0.7158	0.7373	0.7651	0.7895	0.8154	0.8418	0.8683
EigenRank	0.6734	0.6799	0.6972	0.7192	0.7408	0.7634	0.7889	0.8146	0.8407	0.8701
ListRankMF	0.6769	0.6792	0.6989	0.7140	0.7316	0.7532	0.7772	0.8057	0.8368	0.8684
FISMauc	0.6480	0.6681	0.6912	0.7132	0.7363	0.7598	0.7826	0.8086	0.8360	0.8661
FISMrmse	0.6735	0.6868	0.7060	0.7246	0.7475	0.7684	0.7914	0.8164	0.8431	0.8726
LLORMA	0.6794	0.6898	0.7092	0.7264	0.7488	0.7705	0.7950	0.8219	0.8462	0.8738
ListCF	0.6846	0.6897	0.7100	0.7274	0.7500	0.7732	0.7982	0.8243	0.8499	0.8752
LwRec	**0.6930**	**0.6991**	**0.7200**$^{\triangle}$	**0.7422**$^{\blacktriangle}$	**0.7643**$^{\blacktriangle}$	**0.7844**$^{\blacktriangle}$	**0.8059**$^{\blacktriangle}$	**0.8287**	**0.8527**	**0.8801**$^{\triangle}$
Performance for MovieLens-1M										
NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10
CF	0.6214	0.6498	0.6710	0.6954	0.7189	0.7437	0.7708	0.7981	0.8272	0.8589
MF	0.6619	0.6649	0.6802	0.7008	0.7238	0.7483	0.7741	0.8026	0.8322	0.8642
EigenRank	0.6486	0.6571	0.6746	0.6958	0.7190	0.7428	0.7688	0.7966	0.8268	0.8608
ListRankMF	0.7084	0.7078	0.7203	0.7342	0.7532	0.7736	0.7972	0.8225	0.8498	0.8803
FISMauc	0.6784	0.6951	0.7109	0.7315	0.7526	0.7750	0.7983	0.8235	0.8493	0.8770
FISMrmse	0.7157	0.7178	0.7279	0.7440	0.7634	0.7849	0.8071	0.8315	0.8569	0.8847
LLORMA	0.7116	0.7174	0.7303	0.7479	0.7672	0.7878	0.8100	0.8340	0.8587	0.8854
ListCF	0.7204	0.7243	0.7359	0.7504	0.7685	0.7895	0.8136	0.8384	0.8627	0.8876
LwRec	**0.7204**	**0.7281**	**0.7428**$^{\blacktriangle}$	**0.7600**$^{\blacktriangle}$	**0.7777**$^{\blacktriangle}$	**0.7988**$^{\blacktriangle}$	**0.8207**$^{\blacktriangle}$	**0.8436**$^{\blacktriangle}$	**0.8667**$^{\blacktriangle}$	**0.8906**$^{\blacktriangle}$

solves the problem by minimizing a listwise loss function using non-negative matrix factorization. Our experimental results on benchmark datasets show significant performance gains of *LwRec* over state-of-the-art recommender algorithms.

In this study, we have experimented for top k item rankings, for $k = 1$. In the future, we would like to explore the effect on results on using higher value of k. Moreover, we have used column length 10 for the matrices U and G. It would be interesting to see the changes in results on varying this column length.

References

1. Gunawardana, A., Shani, G.: A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. J. Mach. Learn. Res. **10** (2009) 2935–2962
2. Adomavicius, G., Tuzhilin, A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. on Knowl. and Data Eng. **17**(6) (2005) 734–749
3. Huang, S., Wang, S., Liu, T.Y., Ma, J., Chen, Z., Veijalainen, J.: Listwise Collaborative Filtering. In: SIGIR. (2015) 343–352
4. Herlocker, J., Konstan, J.A., Riedl, J.: An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. Inf. Retr. **5**(4) (2002) 287–310
5. Linden, G., Smith, B., York, J.: Amazon.com Recommendations: Item-to-item Collaborative Filtering. IEEE Internet Comput. **7**(1) (2003) 76–80

6. Ning, X., Karypis, G.: SLIM: Sparse Linear Methods for Top-N Recommender Systems. In: ICDM. (2011) 497–506
7. Kabbur, S., Ning, X., Karypis, G.: FISM: Factored Item Similarity Models for top-N Recommender Systems. In: SIGKDD. (2013) 659–667
8. Koren, Y., Bell, R., Volinsky, C.: Matrix Factorization Techniques for Recommender Systems. Computer **42**(8) (2009) 30–37
9. Salakhutdinov, R., Mnih, A.: Probabilistic Matrix Factorization. In: NIPS. (2007) 1257–1264
10. Lee, D.D., Seung, H.S.: Algorithms for Non-negative Matrix Factorization. In: NIPS. (2000) 556–562
11. Rendle, S.: Factorization Machines with libFM. ACM Trans. Intell. Syst. Technol. **3**(3) (2012) 57:1–57:22
12. Gopalan, P., Hofman, J.M., Blei, D.M.: Scalable Recommendation with Hierarchical Poisson Factorization. In: UAI. (2015) 326–335
13. Lee, J., Kim, S., Lebanon, G., Singer, Y.: Local Low-rank Matrix Approximation. In: ICML. (2013) II–82–II–90
14. Liu, N.N., Yang, Q.: EigenRank: A Ranking-oriented Approach to Collaborative Filtering. In: SIGIR. (2008) 83–90
15. Wang, S., Sun, J., Gao, B.J., Ma, J.: VSRank: A Novel Framework for Ranking-Based Collaborative Filtering. ACM Trans. Intell. Syst. Technol. **5**(3) (2014) 51:1–51:24
16. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., Hanjalic, A.: CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-more Filtering. In: RecSys. (2012) 139–146
17. Weimer, M., Karatzoglou, A., Le, Q.V., Smola, A.: CofiRank: Maximum margin matrix factorization for collaborative ranking. In: NIPS. (2007) 1593–1600
18. Pan, W., Chen, L.: GBPR: Group Preference Based Bayesian Personalized Ranking for One-class Collaborative Filtering. In: IJCAI. (2013) 2691–2697
19. Marden, J.I.: Analyzing and Modeling Rank Data. CRC Press (1996)
20. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to Rank: From Pairwise Approach to Listwise Approach. In: ICML. (2007) 129–136
21. Kullback, S.: Information Theory And Statistics. Dover Pubns (1997)
22. Breese, J.S., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: UAI. (1998) 43–52
23. Shi, Y., Larson, M., Hanjalic, A.: List-wise Learning to Rank with Matrix Factorization for Collaborative Filtering. In: RecSys. (2010) 269–272
24. Järvelin, K., Kekäläinen, J.: Cumulated Gain-based Evaluation of IR Techniques. ACM Trans. Inf. Syst. **20**(4) (2002) 422–446